



ACPMAPS:  
The Fermilab Lattice Supercomputer  
Project\*

P. Mackenzie, E. Eichten, G. Hockney, and H. B. Thacker  
*Theoretical Physics Group*  
*Fermi National Laboratory<sup>†</sup>*  
*Batavia, IL 60510 USA*

R. Atac, A. Cook, M. Fischler, I. Gaines, D. Husby, and T. Nash  
*Advanced Computer Program*  
*Fermi National Laboratory<sup>†</sup>*  
*Batavia, IL 60510 USA*

**Abstract**

The ACP Multi-Array Processor System (ACPMAPS) is a highly cost effective, local memory parallel computer designed for floating point intensive grid based problems. The processing nodes of the system are single board array processors based on the FORTRAN and C programmable Weitek XL chip set. The nodes are connected by a network of very high bandwidth 16 port crossbar switches. The architecture is designed to achieve the highest possible cost effectiveness while maintaining a high level of programmability. The primary application of the machine at Fermilab will be lattice gauge theory.

---

\*Talk given by Paul Mackenzie at the International Symposium "Field Theory on the Lattice", Seillac, France, Sept. 28 - Oct. 2, 1987.

<sup>†</sup>Fermilab is operated by Universities Research Association, Inc. under contract with the U. S. Department of Energy



# 1 Introduction

The ACP Multi-Array Processor System (ACPMAPS) is a high performance, cost effective parallel computer designed for floating point intensive grid based problems, primarily lattice gauge theory. To obtain some estimates of the computing needs of lattice quantum chromodynamics one can consider the calculation of the deconfining temperature in SU(3) gauge theory without quarks [1], which is one of the most solid four dimensional calculations done so far. This calculation required a lattice spacing of less than 0.1 fermi and a volume of close to  $(2 \text{ fermi})^3$ , resulting in lattices with spatial sizes of up to  $19^3$ . It is virtually certain that calculations with quarks will require even larger lattices than this for comparable accuracy. Lattices with space-time sizes  $32^4$  to  $64^4$ , requiring 1 - 20 GBytes of data memory, are a reasonable guess. Calculations of hadron masses in the approximation of ignoring dynamical quarks have not yet achieved a reasonable understanding of calculational errors, even on Cray-sized supercomputers. Although algorithms for the inclusion of dynamical quark effects have made tremendous progress in the last few years, at present they still seem to require at least two orders of magnitude more computer time than comparable calculations without quarks. It is thus clear that large increases in combined CPU power *and* algorithmic power are still required for even simple QCD calculations.

The aim for the Fermilab lattice machine is the delivery of these large amounts of memory and CPU power at a reasonable cost, without compromising the programmability required for rapid algorithm development which is just as important as raw computing power in achieving the goals of lattice gauge theory. For other large-scale computer projects aimed at lattice gauge theory, see [2].

# 2 Architecture Overview

A block diagram of the system is shown in Figure 1. A detailed technical description will appear in [3]. The backbone of the system is a network of switch crates (with two shown in detail in the figure) which handle communication between the nodes. These are 16 port crossbar switches which transfer data at rates of 20 MBytes/sec per channel times 8 possible channels per crate. To each switch crate is attached a group of individual nodes. The nodes are single board array processors using the Weitek XL chip set

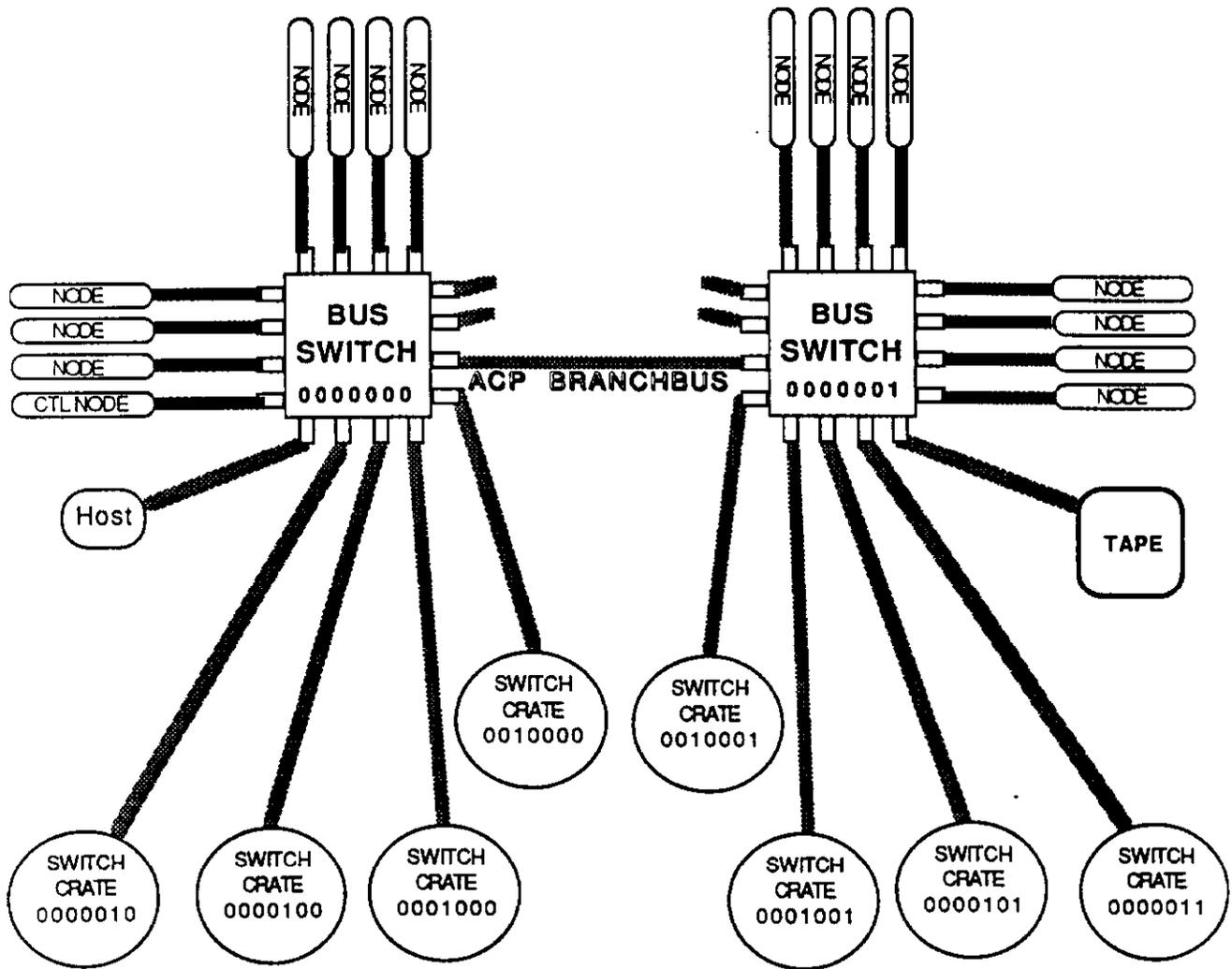


Figure 1: Block diagram of ACPMAPS.

which contains a 32 bit, 20 MFlop (peak) floating point unit, an integer processor, 32 floating point and 32 integer registers, and an instruction sequencer. The chip set as a whole is programmable in FORTRAN and C, at a some sacrifice in performance. Thus, these modules incorporate the functions of a high level language programmable single board computer and a high performance floating point array processor. No external CPU is required as a controller for these stand alone floating point engines.

The system is controlled by a host microVAX. The host starts a control program running on a control node, which is identical to all the other nodes of the system except in software. This node controls the lattice-wide parts of the program and starts subprocesses on the individual nodes.

The nodes operate completely asynchronously. The use of MIMD (multiple instruction, multiple data) rather than SIMD (single instruction, multiple data) architecture is one of the most important features of the system. There are many advantages to this type of architecture. It is very flexible: it can handle problems which are awkward or impossible in SIMD such as heat bath and incomplete LU decomposition algorithms and random lattice problems. The allowed sizes and shapes of the lattices are independent of the details of the hardware. The node structure of the machine can be made invisible in much or all of the high-level user code, resulting in improved programmability. This also results in improved fault tolerance, since the system can be reconfigured with one fewer nodes when one node is down, without requiring changes in user software or the setting aside of any spare nodes. Complications which have to be faced with MIMD include the potential for synchronization conflicts, which can't occur with SIMD. This requires care in designing and understanding the communications system. In addition, a nontrivial system software design effort is required to ensure that overheads associated with the communications software are kept to acceptable levels.

## 3 Hardware

### 3.1 The Nodes

The floating point boards contain the XL chip set, the data and code memory, and the interface logic and input and output queues for communicating with the crossbar switches. One floating point unit is used per node, in contrast to the designs of most of the SIMD machines aimed at lattice gauge

theory. In addition to being a flexible and sensible design for a wide variety of problems, this was dictated by the desire to be able to use the Weitek FORTRAN and C compilers for the XL chip set.

The 2 MBytes of program memory and 8 MBytes of data memory is made from 1 Mbit page mode dynamic RAM chips, which in page mode can deliver data at a rate of one word per 100 nsec. This rate is fast enough that little additional efficiency would be gained in most lattice algorithms by the replacement of some of the DRAM by faster, more expensive static RAM. The memory chips constitute almost a third of the total cost of the machine. The memory to power ratio provided (8 MBytes to 20 MFlops) is larger than that provided by most other machines of this type, and is larger than is required by presently existing algorithms for simulating full QCD including internal fermion loops. It is approximately appropriate for calculations in the valence approximation, ignoring fermion loops. Algorithmic improvements over the next few years will certainly change the required ratio. It seems likely that the possibilities which will increase the required ratio (preconditioning and Fourier acceleration of quark propagator calculation, Fourier acceleration of gauge simulation) are currently more promising than those which reduce the amount of memory required per CPU cycle (such as adding nonlocal operators to the action to reduce finite lattice spacing errors) and that the large amount of memory could easily become crucial in the years to come.

## 3.2 Node-Node Communications

The nodes are plugged into a network of switch crates whose backplanes handle full sixteen port crossbar switching at bandwidths of 20 MBytes/second per connection. This yields a total bandwidth of 2.56 GBytes/sec for a 256 node machine. A cluster of 8 - 12 nodes is attached to each switch. The switches are connected in a hypercube, which may be augmented by additional communications channels along heavily used paths. This structure allows the nodes to communicate as if they were connected in a conventional hypercube arrangement, but more than this, it allows any node to communicate at full speed with *any* other node, allowing efficient running of algorithms requiring nonlocal communications. The switch crates allow any node to access any other node's data memory without needing to know where the other node is located on the network. With the current switch crate hardware, systems of up to 2048 nodes are possible before this transparent nonlocal communication feature is lost. The switch

crates are based on SN74AS8840 sixteen input crossbar switch chips from Texas Instruments. They will also be used in hardware being developed by the ACP for high performance applications of the ACP Multiprocessor System [4]. This system is in use for a variety of experimental particle physics applications, including the "Level-3" programmable trigger for the Collider Detector at Fermilab.

### 3.3 The Tape System

Low cost video technology tape drives are expected to be used for checkpointing long calculations and for archiving of gauge fields and propagators. These tape drives cost a few thousand dollars and can store 2 GBytes of data on an 8 mm video tape cartridge costing \$5 - \$10. One drive will be attached to every second switch crate, enabling all of memory to be stored in under ten minutes.

## 4 Software

Lattice gauge theories are part of a large class of grid-based problems derived from discretization of a set of differential equations which are very suitable for a parallel architecture like this one. The natural breakdown of the problem is to assign a certain subset of the sites in the space or space-time to each node, which stores the data for the field variables defined on the sites assigned to it in its local memory and does calculations for its sites. The system software has been designed to shield the user as much as possible from the hardware dependent node structure of the parallel architecture.

User programs are divided conceptually into two pieces: the control program, which is called from the microVAX host and runs on the control node, and site subroutines, which run on the individual nodes. The control program controls the execution of lattice-wide tasks. It will typically be written in ordinary FORTRAN or C augmented by a set of system subroutines for dealing with global concepts (e. g., field memory, lattice wide tasks) which are distributed over all the nodes and require special treatment. The beginning of the control program will include statements like the following:

```
call define_periodic_lattice(ndims,sizes,lat1)
call define_field( lat1, quarksize, q )
```

```

call define_field( lat1, quarksize, q1 )
call complete_definitions

```

The routine `define_periodic_lattice` tells the system that our problem contains one lattice called `lat1` of `ndims` dimensions with the size of each dimension contained the array `sizes` and with standard hypercube connectivity. More general user defined connectivities will be allowed. It is possible to define several lattices in the same program for block spin renormalization group or multigrid algorithms. The routine `define_field` tells the system that memory will be required for storing two fields identified by `q` and `q1`, each with `quarksize` components for each site of `lat1`. The routine `complete_definitions` calls routines which assign specific sites to specific nodes, allocate memory in the nodes for the field data and site structures, and set up structures for each site pointing to the memory areas of adjacent sites of the lattice.

A control node subroutine which operates on a field `q` with an operator `dslash` and stores the result in another field `q1` would be written as follows.

```

subroutine dslash( q, q1 )
.
.
.
call do_task( dslash_, lat1,
              pass$, q, 1,
              pass$, q1.1,
              end$)

return
end

```

The system subroutine `do_task` passes to all the nodes a pointer to a subroutine `dslash_` which operates on a single site and a pointer to a list of sites on which to operate, which may be the entire lattice `lat1` or some previously defined set of sites such as `red_sites`. A system routine on the node, invisible to the user, calls `dslash_` for all the sites in the set of sites which have been assigned to the node. `Do_task` may also be used to pass (`pass$`) to the nodes parameters required by the site subroutine (like the field identifiers `q` and `q1`) and to integrate (`integrate$`) data returned from the individual nodes.

The site subroutines access and replace data from global fields with subroutines like:

```
call get_field( q, site, qtemp )
call put_field( q1, site, qtemp ),
```

which determine if the desired data is already resident on the node and open a channel to the communications hardware if necessary.

Inessential site subroutines can be written in a high level language. CPU intensive kernels such as SU(3) matrix multiplication and essential routines like dslash will be microcoded for maximum efficiency.

A system software simulator implementing a full set of the proposed system subroutines has been written in C. A Cabibbo-Marinari QCD simulation program written in FORTRAN using this simulator software runs on a VAX with a 20% performance hit (which we expect to reduce to 10%) compared to a highly optimized program written in ordinary FORTRAN.

## 5 Physics Program

The main interest of the Fermilab lattice group is the application of lattice gauge theory to QCD and "beyond the standard model" phenomenology. During the initial phase of running the machine we will pursue a program which as much as possible serves the multiple purposes of machine shake-down, algorithm development, careful error analysis for well established quantities like the hadron spectrum, and production of new physics results.

Enormous progress has been made in the last three years on algorithms for inclusion of dynamical quark loops in QCD calculations. Many (possibly most) groups with large machines have begun attempts to calculate hadron masses with the new algorithms. Since there do not yet exist definitive spectrum results in the approximation of ignoring quark loops, this may seem somewhat premature. It is dictated to some extent by the availability of machines with low memory to CPU power, which cannot easily go to lattices with larger volumes and smaller lattice spacing. With our machine, it will make more sense to do a careful study of the statistical, finite volume, and finite lattice spacing errors in the valence approximation before going on to the inclusion of internal quark loops.

In the valence approximation, CPU time is dominated by the efficiency of the quark matrix inversion algorithm (as opposed to the product of the efficiencies of the quark matrix inversion and the simulation algorithms in dynamical fermion algorithms). The relative efficiencies of these algorithms is very dependant on lattice size, the quark mass, etc. Our first algorithm project will be the testing on large lattices of the most promising of these

methods (conjugate gradient, minimal residual, ...) and various methods of preconditioning (Fourier acceleration, incomplete LU decomposition).

During the initial period of running, we will be doing careful analysis of errors in the valence approximation on a variety of lattice spacings and volumes. We intend to have as complete as possible a set of analysis programs for quantities which can be calculated from the basic set of quark propagators to run as a standard package on all data sets produced. These will include meson and baryon heavy quark potentials, the light hadron spectrum and decay constants, the kaon bag parameter, and the properties of D and B mesons in the  $1/M$  expansion, including masses, spin splittings, decay constants and bag parameters.

## 6 Current Status

An eighteen node production prototype system is currently being assembled, with a production system of 256 nodes anticipated. The 256 node computer will cost well under \$1 million, with a peak speed of over 5 GFlops and a memory of over 2 GBytes. Expansion to 2048 nodes is possible, producing a peak speed of 40 GFlops and a data memory of 16 Gbytes. It is intended that the machine become commercially available, allowing its application to many other problems.

## References

- [1] S. A. Gottlieb et al., Phys. Rev. Lett. **55** 1958 (1985); N. H. Christ and A. E. Terrano, Phys. Rev. Lett. **56** 111 (1986).
- [2] N. H. Christ and A. E. Terrano, IEEE Trans. Comp. **C-33**(4) 344 (1984); C. L. Seitz, Commun. ACM **28** 22 (1985); J. Beteem, M. Denneau and D. Weingarten, J. Stat. Phys., **43** 1171 (1986); M Albanese et al., The APE Computer, ROM2F/87/005 (1987).
- [3] M. Fischler et al., The ACP Multi-Array Processor System, paper submitted to The Third Conference on Hypercube Concurrent Computers and Applications, Pasadena, CA, Jan. 19-20, 1988.
- [4] I. Gaines et al., The ACP Multiprocessor System at Fermilab, Computing in High Energy Physics Conference, Asilomar State Beach, (1987), published in Comp. Phys. Com. **45** 323 (1987).